

# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-287571

(43)Date of publication of application : 31.10.1995

(51)Int.Cl.

G10H 1/00  
G10L 3/00

(21)Application number : 06-081392

(71)Applicant : YAMAHA CORP

(22)Date of filing : 20.04.1994

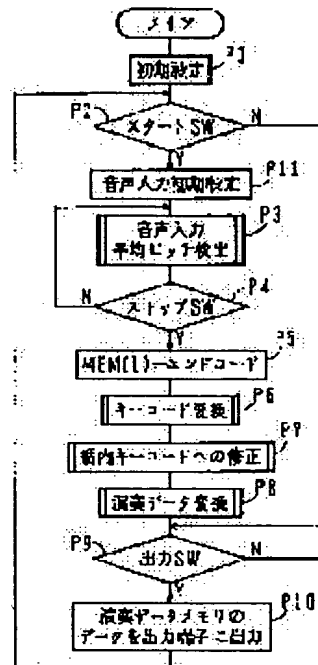
(72)Inventor : AOKI EIICHIRO

## (54) SOUND SIGNAL CONVERSION DEVICE

### (57)Abstract:

**PURPOSE:** To convert a sound signal into a correct key code irrespective of the kind of the inputted sound signal.

**CONSTITUTION:** This sound signal conversion device is provided with a sound input means which inputs sound and generates the sound signal, a pitch extracting means which divides the sound signal generated by the sound input means into specific time sections and extracts pitches in the respective time sections, a mean pitch arithmetic means which calculates the mean pitch of the pitches extracted by the pitch extracting means, and a pitch information converting means which converts the mean pitch calculated by the mean pitch arithmetic means into pitch information. Further, the device has a sound input means which inputs a sound constituting music and generates a sound signal, a pitch extracting means which extracts a pitch from the sound signal generated by the sound input means, a pitch information converting means which converts the pitch extracted by the pitch extracting means into pitch information, a tone information supply mean which supplies tone information, and a pitch information correcting means which corrects the pitch information according to the tone information supplied from the tone information supply means.



## LEGAL STATUS

[Date of request for examination] 27.06.1997

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number] 3055392

[Date of registration] 14.04.2000

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平7-287571

(43) 公開日 平成7年(1995)10月31日

(51) Int.Cl. <sup>8</sup>	識別記号	庁内整理番号	F I	技術表示箇所
G 1 0 H 1/00		B		
G 1 0 L 3/00		B		

審査請求 未請求 請求項の数 2 O L (全 13 頁)

(21) 出願番号 特願平6-81392

(22) 出願日 平成6年(1994)4月20日

(71) 出願人 000004075

ヤマハ株式会社

静岡県浜松市中区町10番1号

(72) 発明者 青木 栄一郎

静岡県浜松市中区町10番1号 ヤマハ株式会社内

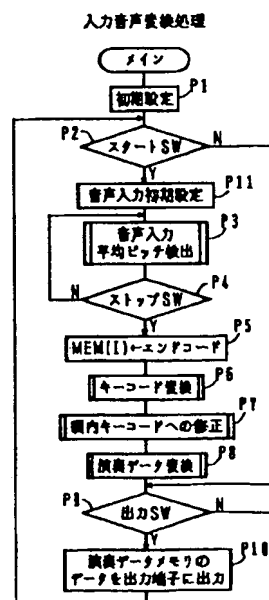
(74) 代理人 弁理士 高橋 敬四郎 (外1名)

(54) 【発明の名称】 音声信号変換装置

(57) 【要約】

【目的】 入力される音声信号の種類にかかわらず、音声信号を正しいキーコードに変換することができる音声信号変換装置を提供することを目的とする。

【構成】 音声を入力して音声信号を生成するための音声入力手段と、音声入力手段が生成する音声信号を所定の時間区間に分割して各時間区間におけるピッチを抽出するためのピッチ抽出手段と、ピッチ抽出手段により抽出される複数のピッチを平均化して平均ピッチを演算するための平均ピッチ演算手段と、平均ピッチ演算手段により演算される平均ピッチを音高情報に変換するための音高情報変換手段とを有する。また、楽曲を構成する音声を入力して音声信号を生成するための音声入力手段と、音声入力手段が生成する音声信号からピッチを抽出するためのピッチ抽出手段と、ピッチ抽出手段により抽出されるピッチを音高情報に変換するための音高情報変換手段と、調情報を供給するための調情報供給手段と、調情報供給手段から供給される調情報に応じて音高情報を修正するための音高情報修正手段とを有する。



## 【特許請求の範囲】

【請求項1】 外部から供給される音声を入力して音声信号を生成するための音声入力手段と、  
前記音声入力手段が生成する音声信号を所定の時間区間に分割して該各時間区間におけるピッチを抽出するためのピッチ抽出手段と、  
前記ピッチ抽出手段により抽出される複数のピッチを平均化して平均ピッチを演算するための平均ピッチ演算手段と、  
前記平均ピッチ演算手段により演算される平均ピッチを音高情報に変換するための音高情報変換手段とを有する音声信号変換装置。

【請求項2】 外部から楽曲を構成する音声を入力して音声信号を生成するための音声入力手段と、  
前記音声入力手段が生成する音声信号からピッチを抽出するためのピッチ抽出手段と、  
前記ピッチ抽出手段により抽出されるピッチを音高情報に変換するための音高情報変換手段と、  
調情報を供給するための調情報供給手段と、  
前記調情報手段から供給される調情報に応じて前記音高情報を修正するための音高情報修正手段とを有する音声信号変換装置。

## 【発明の詳細な説明】

## 【0001】

【産業上の利用分野】 本発明は音声信号変換装置に関し、特に入力された音声信号をキーコードに変換する変換装置に関する。

## 【0002】

【従来の技術】 音声信号変換装置は、外部から入力される音声信号をキーコード（音高情報）に変換する装置である。変換されたキーコードは、例えばMIDI形式の演奏データとして扱うことができる。音源装置でこのような演奏データを楽音信号に変換すれば、発音可能な信号となる。

【0003】 音声信号変換装置は、逐次入力される音声信号が何のキーコードであるのかを検出するための認識処理を行う。音声信号から正しいキーコードを認識することができる割合を示すキーコード認識率は、音声信号変換装置に入力される音声信号の種類により大きく異なる。

【0004】 入力音声信号が楽器演奏により発せられたものであれば、キーコードの認識率はある程度高い値がえられる。しかし、入力音声信号が人間の音声である場合には、楽器演奏による音声の検出に比べてキーコードの認識率が低い値になってしまう。

【0005】 人間の喉から発せられる声は、通常同一のキーコードを持続して有することは少なく、揺らいだ音になる。プロ歌手による歌声であっても、発せられる音声信号中の基本周波数はある程度揺れてしまうために、歌声から1音1音のキーコードを正確に認識することは

困難である。

## 【0006】

【発明が解決しようとする課題】 電子楽器のように、電気信号の合成により生成される音声信号は、各周波数成分の変動が少なく雑音がほとんど含まれていない信号である。したがって、電子楽器による音声信号から対応するキーコードを認識することは比較的容易に行うことができる。

【0007】 しかし、人間の発声等により発せられる音声信号は、周波数成分の変動が大きいためにキーコードを認識することが困難である。本発明の目的は、入力される音声信号の種類にかかわらず、音声信号を正しいキーコードに変換することができる音声信号変換装置を提供することである。

## 【0008】

【課題を解決するための手段】 本発明の音声信号変換装置は、外部から供給される音声を入力して音声信号を生成するための音声入力手段と、音声入力手段が生成する音声信号を所定の時間区間に分割して各時間区間におけるピッチを抽出するためのピッチ抽出手段と、ピッチ抽出手段により抽出される複数のピッチを平均化して平均ピッチを演算するための平均ピッチ演算手段と、平均ピッチ演算手段により演算される平均ピッチを音高情報に変換するための音高情報変換手段とを有する。

【0009】 また、本発明の音声信号変換装置は、外部から楽曲を構成する音声を入力して音声信号を生成するための音声入力手段と、音声入力手段が生成する音声信号からピッチを抽出するためのピッチ抽出手段と、ピッチ抽出手段により抽出されるピッチを音高情報に変換するための音高情報変換手段と、調情報を供給するための調情報供給手段と、調情報供給手段から供給される調情報に応じて音高情報を修正するための音高情報修正手段とを有する。

## 【0010】

【作用】 ピッチ抽出手段は、音声信号入力手段に入力された音声信号を所定の時間区間に分割して、各時間区間毎のピッチを検出する。検出されたピッチは、平均ピッチ演算手段により複数の時間区間で平均化された後に、音高情報に変換される。音高情報は、検出されたピッチを直接用いるよりも、平均化したピッチを用いた方が正しい音高情報を得ることができる。

【0011】 また、楽曲を構成する入力音声信号から検出されたピッチを音高情報に変換した後に、音高情報は、供給された調情報の音楽規則に応じて修正される。音高情報の修正を行うことにより、正しい音高情報を得ることができる。

## 【0012】

【実施例】 図1は、本発明を実施するための音声信号変換装置のシステム構成例を示す。音声信号変換装置は、マイク1から入力される音声信号をキーコードに変換す

ることができる。変換されたキーコードは、音源4において対応するキーコードの楽音信号に変換され、サウンドシステム5から発音される。

【0013】マイク1には、外部から歌声等の音声信号が入力される。入力される音声信号は、例えば人の歌声や楽器演奏等による楽音のアナログ信号である。スタートスイッチを押すと、音声信号の入力が開始され、ストップスイッチを押すと、音声信号の入力が終了する。マイク1に入力された音声アナログ信号は、D/A変換器2においてデジタル信号に変換され、バス11に供給される。

【0014】CPU7は、バス11に供給された音声デジタル信号をキーコードに変換するために、各種の制御処理を行う。正しいキーコードに変換するために、CPU7は通常のキーコード変換処理の他に、平均ピッチ検出と調内キーコードへの修正処理を行う。

【0015】平均ピッチ検出は、マイク1に入力された音声信号を量子化して、量子化された音声信号毎にピッチを検出する。そして、所定時間内で検出された複数のピッチの平均ピッチを求め、平均ピッチからキーコードを算出する。平均ピッチを検出することにより、人の歌声のように時間経過と共にピッチが変化しやすい性質の音声信号であっても、正しいピッチを検出することができる。

【0016】調内キーコードへの修正処理は、まず、連続してマイク1に入力された音声信号を楽曲とみなして、その楽曲の調を検出する。楽音の調が検出されれば、その調に用いられやすいキーコードを予測することができる。この音楽規則の知識を利用して、検出されたキーコードの修正処理を行う。これにより、誤認識により検出されたキーコードを正しいキーコードに修正することができる。

【0017】データメモリ(RAM)10は、入力音声信号を楽音とみなしたときの調を検出するために必要なデータ等を記憶している。例えば、C長調、C短調等の各調を構成する音名データを記憶しており、入力音声信号のキーコードがどの調の構成音に該当するのかを調べることににより、調の検出が行われる。

【0018】CPU7は、入力音声信号からキーコードに変換した後に、変換したキーコードを基にして演奏データを生成する。生成された演奏データは音源4に供給され、楽音信号が生成される。サウンドシステム5は、音源4から供給される楽音信号に応じて楽音を発する。

【0019】また、出力端子3の出力スイッチが押されると、生成された演奏データは、必要に応じてMIDI形式のデータ等に変換され、出力端子3から外部機器に出力される。

【0020】プログラムメモリ(ROM)8は、演算プログラムを記憶している。CPU7はこの演算プログラムに従って、ワーキングメモリ(RAM)9に備えられ

たレジスタやバッファメモリ等のワーキングメモリを用いて上述の各種演算処理を行う。

【0021】タイマ6は、タイミング信号を生成し、所定時間間隔でCPU7に割り込み信号を供給する。タイマ6は、例えば16分(音符)の長さ毎に割り込み信号をCPU7に供給し、割り込み処理を実行させる。割り込み処理は、外部から人がマイク1に歌声を入力する際等に必要となるメトロノーム音を生成することもできる。

【0022】CPU7は、バス11を介して、D/A変換器2、プログラムメモリ8、ワーキングメモリ9、データメモリ10、音源4、出力端子3の制御を行う。図2は、CPUが行う入力変換処理のメインルーチンを示すフローチャートである。CPUは、まずステップP1において、レジスタ類の初期化等の初期設定の処理を行う。

【0023】ステップP2では、マイクから音声信号の入力開始を指示するスタートスイッチが押されたか否かを調べる。スタートスイッチが押されていないければ、音声入力の処理を行わずにステップP9へ進む。

【0024】スタートスイッチが押されていれば、ステップP11へ進み、音声入力処理のために必要なレジスタやフラグ等の初期設定を行い、ステップP3へ進む。ステップP3では、音声入力処理と平均ピッチ検出処理を行う。音声入力処理は、マイクから入力される音声信号をワーキングメモリに取り込む処理である。平均ピッチ検出は、入力された音声信号を量子化して、量子化された音声信号毎のピッチの平均値を検出する処理である。処理の詳細は、後にフローチャートを参照しながら説明する。

【0025】ステップP4では、マイクから音声信号の入力終了を指示するストップスイッチが押されたか否かを調べる。ストップスイッチが押されていないければ、マイクに逐次入力される音声信号について、上述の音声入力処理と平均ピッチ検出処理を繰り返し、ストップスイッチが押されるまで処理を繰り返す。

【0026】ストップスイッチが押されると、ステップP5へ進み、音声信号の入力終了を示すエンドコードをワーキングメモリ中のメモリMEM()に格納する。上述のステップで検出された所定時間毎の平均ピッチは、既に配列メモリMEM()に格納されている。入力された音声信号の全ての平均ピッチが格納された後に、ストップスイッチが押されると、最後の配列のメモリMEM(I)にエンドコードが格納される。

【0027】ステップP6では、メモリMEM()に格納された音声信号の平均ピッチをキーコードに変換する処理を行う。平均ピッチは、周波数で表されているので、その周波数で表される値をC4等で表されるキーコードに変換する。詳細は、後述する。

【0028】ステップP7では、変換されたキーコード

の中で誤って変換されたキーコードの修正を行うための処理を行う。まず、前ステップにおいて得られた複数のキーコードを楽曲とみなして、その楽音の調を検出する。そして、検出された調の構成音に該当しないものについては、誤ったキーコードに変換されたと判断して、修正処理を行う。処理の詳細は、後にフローチャートを参照しながら説明する。

【0029】ステップP8では、キーコードの検出および修正が行われた後に、そのキーコードを基にして演奏データの生成を行う。演奏データは、キーコードの他に、発音開始を示すキーオンコード、発音の持続時間を示すデュレーションコード、発音終了を示すキーオフコード等のデータを含んでいる。詳細は、後述する。

【0030】ステップP9では、出力スイッチが押されたか否かを調べる。出力スイッチが押されていれば、ステップP10へ進み、生成された演奏データを出力端子から出力する。その後、ステップP2へ戻り、再び音声入力開始のためのスタートスイッチが押されたか否かを調べる。

【0031】ステップP9において、出力スイッチが押されていないと判断されたときには、出力端子から演奏データを出力することなく、直接ステップP2へ戻る。図3は、図2のステップP3における音声入力処理と平均ピッチ検出処理の詳細を示すフローチャートである。

【0032】図2のメインルーチンにおいて、この音声入力処理が行われる前に、ステップP11で音声入力処理のための初期設定が既に行われている。初期設定では、キーオンフラグKON、ピッチレジスタF、フラグQ、カウンタI、カウンタNの初期化を行っている。各フラグ等の初期設定方法を次に示す。

【0033】まず、キーオンフラグKONを0にする。キーオンフラグKONは、マイクから何らかの音声信号が入力されている間は1を示し、入力される音声信号がほぼ0である間は0を示す。

【0034】次に、ピッチレジスタFを0にクリアする。ピッチレジスタFは、マイクから入力される音声信号から検出されるピッチを格納するためのレジスタである。ピッチとは、音高を表す周波数である。

【0035】そして、フラグQを0にする。フラグQは、タイマからの割り込み信号をきっかけとする割り込み処理により、所定時間（例えば16分音符の長さ）間隔で1にセットされる。フラグQは、所定の処理が行われた後に0になるので、通常は0がセットされており、所定時間間隔で一瞬1がセットされる。

【0036】さらに、カウンタIおよびカウンタNを0にリセットする。カウンタIは、求められる平均ピッチを格納する配列メモリMEM()の配列番号を示す。カウンタNは、所定時間内に入力される音声信号の量子化数をカウントする。

【0037】以上の初期設定が行われた後に、音声入力

および平均ピッチ検出を行うための処理がステップQ2から開始する。ステップQ2では、キーオンフラグKONが0であるか否かを調べる。キーオンフラグKONが0であれば、ステップQ3に進む。最初は、初期設定でキーオンフラグKONが0に設定されているので、ステップQ3に進む。

【0038】ステップQ3では、マイクから入力される音声信号の入力レベルがしきい値#1を越えているか否かを調べる。しきい値#1は、音声入力レベルの雑音レベルを示す。しきい値#1を越えていなければ、入力が無音状態であると判断して、入力音声信号の処理を行わずにステップQ10へ進む。

【0039】音声入力レベルがしきい値#1を越えていれば、適正な音声入力があったとして、ステップQ4へ進み、入力音声信号の処理を開始する。ステップQ4では、適正な音声入力の開始を示すために、キーオンフラグKONを1にする。その後、ステップQ5へ進む。

【0040】ステップQ5では、入力音声信号を基にしてピッチ検出を行う。デジタル信号系列で表される入力音声信号に対して、ハミング窓やハニング窓等で所定時間内の信号を切り出し、周波数解析を行う。周波数解析を行うことにより、ピッチ（音高情報）を検出する。

【0041】ステップQ6では、検出されたピッチをピッチレジスタFに加算する。ピッチレジスタFには、初期設定で0にセットされているので最初は、検出されたピッチそのものの値がピッチレジスタFに格納される。

【0042】ステップQ7では、カウンタNをインクリメントする。その後、ステップQ10へ進む。ステップQ10では、フラグQが1であるか否かを調べる。フラグQは、初期設定において0にセットされているので、一旦図2のメインルーチンの処理へ戻る。もし、音声入力の終了を指示するストップスイッチが押されていなければ、再び上述の音声入力処理を繰り返すためにステップQ2からの処理を再開する。ストップスイッチが押されるまで、音声入力処理を繰り返すことにより、マイクから逐次入力される音声信号を時間経過と共にリアルタイムに処理することができる。

【0043】音声入力処理を再開するために、ステップQ2において、再びキーオンフラグKONが0であるか否かを調べる。既に上述の1回目のステップQ4の処理でキーオンフラグKONが1にセットされているので、ステップQ8へ進む。

【0044】ステップQ8では、今回マイクから入力された音声入力レベルがしきい値#2よりも小さいか否かを調べる。しきい値#2は、音声入力レベルの雑音レベルを示す。しきい値#2を越えていれば、前回から適正な音声信号が引き続き入力されていることを示す。一方、しきい値#2を越えていなければ、マイクからの入力が無音状態になったことを示す。

【0045】ステップQ8において、音声入力レベルが

10

20

30

40

50

しきい値#2を越えていればマイクから音声入力引き続き行われていることを示すので、ステップQ5へ進み、今回入力された音声信号の平均ピッチ検出の処理を行う。

【0046】ステップQ5では、入力音声信号のピッチを検出し、検出したピッチをステップQ6においてピッチレジスタFに加算する。ステップQ7では、カウンタNをインクリメントする。

【0047】ストップスイッチが押されるまでの間は、ピッチレジスタFにマイクから逐次入力される音声信号のピッチが累算されていく。ピッチが累算される度にカウンタNはインクリメントされるので、カウンタNにはピッチレジスタFにピッチが累算された数が格納される。

【0048】その後ステップQ10において、フラグQが1か否かを調べる。フラグQは、次に説明する割り込み処理により値が決定される。図4は、CPUが行う割り込み処理を示すフローチャートである。CPUは、タイマから例えば16分長（16分音符の時間長）の所定の時間間隔で割り込み信号を受ける。CPUは、割り込み信号を受けると以下の割り込み処理を行う。ここで、割り込み処理は16分長間隔で行われるとする。

【0049】まず、ステップR1においてフラグQを1にセットする。そして、ステップR2において、クリック音を発生する。その後、割り込み処理を終了して、元の音声入力の処理を再開する。

【0050】以上のように、割り込み処理では16分長間隔でフラグQを1にセットして、クリック音を発生する。クリック音は、メトロノームの役割を果たす。マイクから音声信号を入力する際には、16分長毎に鳴るクリック音のテンポに合わせて入力を行えばよい。

【0051】図3のステップQ10では、16分長が経過した時にフラグQが1であると判断され、ステップQ11へ進む。ステップQ11では、カウンタNが0より大きいのか否かを調べる。カウンタNには、16分長の間に量子化してピッチを検出した数が格納されている。もし、マイクからの入力音声信号が16分長の間ずっと無音状態であると判断されたときには、カウンタNは0を示す。

【0052】上述のように、マイクに適正な音声入力があるとき、カウンタNには1以上の数が格納されているので、ステップQ12へ進む。ステップQ12では、次式により平均ピッチを求め、平均ピッチレジスタAFに格納する。

$$【0053】AF = F / N$$

ピッチレジスタFには、16分長の間に検出されたピッチの累算値が格納されている。ピッチの累算値Fを量子化数Nで割ることにより、ピッチの平均値AFを求めることができる。その後、ステップQ14へ進む。

【0054】一方、ステップQ11において、16分長

の間無音の音声入力状態が続いたときにはカウンタNが0であるので、ステップQ13へ進む。ステップQ13では、平均ピッチレジスタAFにオール1を格納する。その後、ステップQ14へ進む。オール1とは、レジスタを構成するビットを全て1にするような値を示す。平均ピッチレジスタAFにオール1が格納されていれば、16分長の無音状態である16分休符がマイクから入力されたことになる。

【0055】ステップQ14では、求められた平均ピッチAFを配列メモリMEM(I)に格納する。カウンタIは、初期設定にて0に設定されているので、平均ピッチAFは0番目の配列番号のメモリMEM(0)に格納される。カウンタIは、求められた平均ピッチの番号を示し、時間経過と共に0から順番に増加していく。

【0056】ステップQ15では、カウンタIをインクリメントして、次の平均ピッチの配列番号をセットする。ステップQ16では、ピッチレジスタF、カウンタN、フラグQのそれぞれに0をセットして、次の平均ピッチを求めるための初期設定を行う。その後、図2のメインルーチンの処理に戻り、ストップスイッチが押されるまで、以上の平均ピッチを求める処理を繰り返し行う。

【0057】次に、マイクからの入力音声が無音状態になったときの処理を説明する。ステップQ2において、前回時に適正な音声入力があれば、キーオンフラグKONは1になっているので、ステップQ8へ進む。音声入力レベルがしきい値#2よりも小さい値になっていれば、ステップQ8で無音状態であると判断され、ステップQ9へ進む。ステップQ9では、キーオンフラグKONを0にして、ピッチ検出を行わないでステップQ10へ進む。ステップQ10では、前述のように16分長毎の平均ピッチを求める処理を行う。

【0058】以上のように、マイクから適正な音声入力があったときには、16分長の範囲内で入力音声信号の量子化を行い、量子化された信号毎のピッチを検出する。そして、検出されたピッチの累算値Fを量子化数Nで割ることにより、平均ピッチAFを求める。

【0059】人の歌声等は、音声信号の揺れが大きいので、音声信号の局所的なピッチを検出するとピッチが頻繁に変化し、ピッチの誤認識が生じやすい。そこで、入力された音声信号を量子化して、量子化毎のピッチを検出した後に平均ピッチを求めることにより、正しいピッチを検出することができる。求められた平均ピッチは配列メモリMEM(I)に格納される。

【0060】図5は、メモリMEM(I)の構成例を示す。メモリMEM(I)には、時間経過と共に求められた平均ピッチが順番に格納されている。メモリ領域15には、求められた平均ピッチを示す周波数が格納されている。平均ピッチは、16分長の間に適正な音声入力があったときに求められるピッチの平均値である。

【0061】メモリ領域16には、オール1のデータが格納されている。オール1のデータは、無音の音声入力16分長の間中続いたときに格納されるデータであり、16分休符を示す。

【0062】メモリMEM () に格納されている周波数は、16分音符のピッチを示す周波数である。ただし、オール1が格納されているときには、16分音符のピッチを示すのではなく16分休符を意味する。

【0063】エンドコード17は、ストップスイッチが押されて、全ての音声入力が終了した後に、図2のステップP5においてメモリMEM () 中の最後の領域に格納される。エンドコード17は、メモリMEM () に連続して格納されている平均ピッチのデータの終了であることを示す。

【0064】図6は、図2のメインルーチン中のステップP6で行うキーコード変換の処理の詳細を示すフローチャートである。キーコード変換は、メモリMEM () に格納されている平均ピッチの周波数からキーコードへの変換を行う処理である。

【0065】ステップS1では、平均ピッチ周波数が格納されているメモリMEM () の中において最低の周波数を示す平均ピッチを抽出し、最低周波数レジスタFminにその最低周波数を格納する。

【0066】ステップS2では、メモリMEM () に格納されている周波数を読み出し、レジスタFに格納する。そして、次式の計算により相対キーコードKCを求める。

$$KC = 12 \log_2 (F / F_{min})$$

相対キーコードKCは、通常のセントの1/100の値を示すものであり、最低周波数Fminを基準にして、周波数Fがどれだけ高いのか又は低いのかを示す。周波数Fが最低周波数Fminと同じ値であれば相対キーコードKCは0になり、周波数Fが最低周波数Fminよりも半音高ければ相対キーコードKCは1になる。

【0067】求められた相対キーコードKCは、周波数が元々格納されていたメモリMEM () に再び格納される。同様にして、メモリMEM () に格納されている全ての周波数を相対キーコードKCに変換する。メモリMEM () には、最低周波数をFminを基準とした1/100のセント表現が格納される。

【0068】ステップS3では、最低周波数Fminをその周波数に最も近いキーコードに変換し、最低周波数のキーコードをレジスタKCminに格納する。例えば、周波数が440 [Hz] であったとすればA4のキーコードに変換して、レジスタKCminに格納する。

【0069】ステップS4では、メモリMEM () に格納されている相対キーコードを読み出し、レジスタKCに格納する。そして、次式の計算によりキーコードKを求める。

$$K = KC + KC_{min}$$

キーコードKは、基準となる最低キーコードKCminに相対キーコードKCを加算することによりえられる絶対キーコードである。

【0071】求められたキーコードKは、相対キーコードが元々格納されていたメモリMEM () に再び格納される。同様にして、メモリMEM () に格納されている全ての相対キーコードを絶対キーコードKに変換する。メモリMEM () には、16分長毎のキーコードが格納される。その後、図2のメインルーチンの処理に戻る。

【0072】図7は、図2のメインルーチン中のステップP7の調内キーコードへの修正処理の詳細を示すフローチャートである。ステップT1では、マイクから連続して入力される音声信号の集合を楽曲とみなして、その楽曲の調を検出する。データメモリには、調の構成音を示す音名データが格納されている。

【0073】図8は、データメモリに格納されている調の構成音の音名データを示す。調は、12音名(C, C#, ..., B)をそれぞれ主音としたときの長調および短調がそれぞれあり、それぞれの調についてのデータがデータメモリに格納されている。なお、より細かい調に分けて格納してもよい。

【0074】例えば、C長調のメモリ領域21には、C長調の構成音であるC, D, E, F, G, A, Bのコードが格納されている。同様に、C#長調のメモリ領域22には、C#長調の構成音のコードが格納され、C短調のメモリ領域23には、C短調の構成音のコードが格納されている。

【0075】調検出は、メモリMEM () に格納されているキーコードが上記のデータメモリに記憶されている調の中でどの調に一番合致するかを調べる。メモリMEM () のキーコードとデータメモリ中の調の構成音を比較して、最も一致度の高い調を入力音声信号の調とする。

【0076】主音レジスタTNには、検出された調の主音(C, C#, D等)を格納する。モードレジスタMDには、検出された調のモード(長調または短調)を格納する。調検出が行われた後、ステップT2へ進む。

【0077】なお、調検出の方法は、上述の方法に限られず他の方法により調検出を行ってもよい。ステップT2では、検出された調を用いて、キーコードを構成音に丸め込む処理を行う。入力音声信号に対応するキーコードが検出された調の構成音ではない場合、キーコードを構成音のキーコードに正す。処理の詳細な説明を次に示す。

【0078】図9は、図7のステップT2の調内音丸め込み処理の詳細を示すフローチャートである。まず、基準となる楽曲の最後の音の丸め込みを行い、その後に残りの音の丸め込みを行う。その処理手順を以下に説明する。

【0079】ステップU1では、メモリMEM () のm

番目（メモリMEM () のキーコードに順についている、1番最後に格納されているキーコードの番号）のキーコードを読み出す。

【0080】ステップU2では、読み出したキーコードが調内音（検出された調の構成音）であるか否かを調べる。調内音であれば、正しく検出されたキーコードであると判断して、ステップU6へ進み、読み出したキーコードそのものをレジスタKCに格納する。その後、ステップU7へ進む。

【0081】一方、読み出したキーコードが調内音でなければ、誤認識により検出されたキーコードであると判断して、ステップU3へ進む。ステップU3では、読み出されたm番目のキーコードを調内音に丸め込む処理を行う。まず、読み出されたキーコードをそのキーコードに最も近い調内音に変換する。もし、同じ位の近さの調内音が複数あれば、次の優先順位に従った調内音に変換する。

【0082】

1度>5度>3度>2度>6度>7度>4度

例えば、検出された調がC長調であるとする。1度の音がCであり、5度の音がGであり、3度の音がEである。1度の音が最優先であり、続いて5度、3度、・・・の順の調内音に変換される。

【0083】楽曲の最後の音は、一般的にその調の1度の音で終わり易いという規則がある。したがって、1度の音を最優先し、続いて最後の音に成りやすい5度、3度、・・・の順の音を優先させて変換を行う。そして、ステップU4において、調内音に変換されたキーコードをレジスタKCに格納する。

【0084】ステップU5では、レジスタKCに格納されているキーコードを元のキーコードを読み出した配列番号のメモリMEM () に記憶し直す。レジスタKCには、調内音に修正されたキーコードが格納されている。ただし、検出されたキーコードが調内音であるときには、正しい認識が行われたとして修正を行わずに検出されたキーコードそのものがレジスタKCに格納されている。

【0085】ステップU7では、番号レジスタmをデクリメントする。番号mの値を1小さくすることにより、時間的に1つ前の音を表す番号mに変えることができる。番号mが示す1つ前の音についての処理を次に行う。

【0086】ステップU8では、メモリMEM () に格納されているm番目のキーコードを読み出す。読み出されたキーコードは最後から2番目の音である。ステップU9では、読み出したキーコードが調内音であるか否かを調べる。調内音であれば、正しく検出されたキーコードであると判断して、メモリMEM () のm番目のキーコードを修正せずにステップU14へ進み、レジスタKCにキーコードを格納した後にステップU13へ進む。

【0087】一方、読み出したキーコードが調内音でなければ、誤認識により検出されたキーコードであると判断して、ステップU10へ進む。ステップU10では、読み出したm番目のキーコードを調内音に丸め込む処理を行う。

ステップU10では、読み出したキーコードをそのキーコードに最も近い調内音に変換する。もし、同じ位の近さの調内音が複数あるときには、レジスタKCに格納されているキーコードに最も近い調内音に変換する。

10 【0088】レジスタKCには、ステップU4またはステップU6のいずれかで格納されたキーコードが記憶されている。つまり、楽曲の最後の音の修正後のキーコードが記憶されている。その最後の音のキーコードに最も近いキーコードに優先的に変換される。

【0089】ステップU11では、調内音に修正変換されたキーコードをレジスタKCに格納し、ステップU12でレジスタKCに格納されたキーコードをメモリMEM () の元の記憶場所に記憶し直す。

【0090】ステップU13では、番号mが1であるか否かを調べる。番号mが1であるということは、楽曲の1番最初の音（最初にマイクから入力された音声信号の音）についての調内音への修正処理を終了したことを示す。修正処理は、楽曲の最後の音から最初の音まで順番に行っているため、最初の音についての処理が終了すれば、全ての音についての処理が終了したことになる。

【0091】番号mが1でなければ、ステップU7へ戻り、番号mをデクリメントして1つ前の音のキーコードについての修正処理を繰り返す。メモリMEM () のm番目のキーコードを読み出し、読み出したキーコードが調内音であればキーコードの修正を行わない。読み出したキーコードが調内音でなければ、最も近い調内音のキーコードに変換を行う。近い調内音が複数あるときには、1つ前の音の修正後のキーコードに近いキーコードに優先的に変換する。

【0092】楽曲の最後の音の優先順位は、前述のように調の1度、5度の音等を基準にして決めたが、最後の音以外の音の優先順位は、1つ前の音のキーコードに近い音に優先して、キーコードを決める。最後の音のみは、調に関連した音になり易いからである。

40 【0093】以上の修正処理により、誤認識されたキーコードが修正されるとメモリMEM () に格納される。修正処理が終了すると、図7の元の処理に戻る。なお、以上のキーコードの修正処理では、全ての調外音を調内音に修正する場合について述べたが、必ずしも全てを修正する必要はない。例えば、1小節内でのみ効力を有する#、b等の臨時記号を用いて部分的に調外音を用いた場合もある。そのような場合には、検出された調において特に使用が禁止されている音（音名）のみを禁止されていない音に修正するようにしてもよい。

50 【0094】また、調を用いてキーコードを修正する他



に、スケールの検出を行い使用する音高の幅を複数検出し、検出された音高の幅より外の音については所定の音高幅内の音に修正する処理を行ってもよい。スケールは調性と和音によって定まり、スケールによって使用される音名が決まるので、この音名に対応する音高にそれぞれ幅を持たせることにより、使用される音高の幅が求められる。

【0095】図10は、図2のメインルーチン中のステップP8の演奏データ変換処理の詳細を示すフローチャートである。演奏データ変換は、入力音声信号から検出されたキーコードを基にして、発音可能な演奏データ形式への変換を行う。

【0096】ステップV1では、配列番号レジスタP、デュレーションレジスタD、キーオンフラグKONにそれぞれ0をセットする。配列番号レジスタPは、キーコードが格納されているメモリMEM ( ) の配列番号を示す。デュレーションレジスタDは、同一のキーコードを発音し続ける持続時間または発音を行わない休符状態の時間を格納する。キーオンフラグKONは、何らかのキーコードを発音中であるときに1となり、いずれのキーコードをも発音しないときには0となる。

【0097】ステップV2では、演奏データメモリ中の書き込み位置を示す書き込みポインタを演奏データメモリの先頭にセットし、最初の演奏データの書き込みの準備を行う。

【0098】ステップV3では、メモリMEM (P) にエンドコードが格納されているか否かを調べる。配列番号Pは、初期設定で0に設定されている。メモリMEM ( ) には、入力音声信号から検出された後に必要に応じて修正が行われたキーコードが格納されている。もし、メモリMEM ( ) にエンドコードが格納されていれば、その配列番号の位置で音声入力終了していることを示す。

【0099】メモリMEM (P) にエンドコードが格納されていなければ、ステップV4へ進み、メモリMEM (P) に格納されているキーコードを基にして演奏データに変換する。配列番号Pについての演奏データ変換方法は、後に説明する。

【0100】メモリMEM (P) のキーコードについて演奏データが生成された後には、ステップV5へ進み、配列番号レジスタPをインクリメントして次の配列番号についての演奏データの生成に備える。また、デュレーションレジスタDをインクリメントして発音の持続時間をカウントする。デュレーションレジスタDが1増えると、16分長の長さだけ持続時間が増加する。メモリMEM ( ) には、16分長毎にキーコードが格納されているからである。

【0101】その後、ステップV3へ戻り、次の配列番号Pに格納されているキーコード (MEM (P) ) がエンドコードであるか否かを調べる。エンドコードでな

れば、ステップV4へ進み、メモリMEM (P) に格納されているキーコードを基にして演奏データの生成を行う。そして、ステップV5において、配列番号レジスタPとデュレーションレジスタDをそれぞれインクリメントする。

【0102】デュレーションレジスタDには、同一のキーコードが連続してメモリMEM ( ) に格納されている数が記憶される。例えば、A4のキーコードが1つだけ現れて次にはキーコードが変わってしまうときには、デュレーションレジスタDに1が格納され、A4の16分音符を示す。また、A4のキーコードが2つ連続して続けば、デュレーションレジスタDには2が格納され、A4の8分音符を示す。

【0103】ステップV3において、メモリMEM (P) にエンドコードが格納されていると判断されたときには、キーコードの終了を示すので、ステップV6へ進む。ステップV6では、キーオンフラグKONが1であるか否かを調べる。キーオンフラグKONが1であれば、現在いずれかのキーコードを発音中であることを演奏データが示しているので、ステップV7へ進む。

【0104】ステップV7では、現在発音中のキーコードについての発音持続時間を示すデュレーションレジスタDの値をデュレーションコードと共に演奏データメモリに書き込む。デュレーションコードは、演奏データメモリにおいて、発音持続時間を示すデュレーションレジスタDのデータが次に続くことを示すコードであり、デュレーションコードとデュレーションレジスタDのデータは、必ず組で演奏データメモリに書き込まれる。書き込みを終了した後、演奏データメモリへの書き込みポインタを進めて次の書き込みの準備をする。

【0105】ステップV8では、発音の終了を示すキーオフコードと演奏データの終了を示すエンドコードを演奏データメモリに書き込む。書き込みを終了した後、演奏データの生成を終了して図2のメインルーチンの処理に戻る。

【0106】ステップV6において、キーオンフラグKONが1でないと判断されたときには、現在発音中ではないことを示すので、ステップV9へ進み、エンドコードのみを演奏データメモリに書き込む。書き込みを終了した後、図2のメインルーチンの処理へ戻る。

【0107】図11は、図10のステップV4における演奏データ変換処理の詳細を示すフローチャートである。メモリMEM (P) にエンドコードではなく、キーコードが格納されているときに、メモリMEM (P) のキーコードを基にして演奏データに変換する手順を示す。

【0108】ステップW1では、メモリMEM (P) にオール1が格納されているか否かを調べる。オール1が格納されていれば、入力が無音状態であることを示す。メモリMEM (P) にオール1が格納されていなければ

10

20

30

40

50

ば、何らかのキーコードが記憶されているので、そのキーコードを基にした演奏データを生成するためにステップW2へ進む。

【0109】ステップW2では、キーオンフラグKONが1であるか否かを調べる。キーオンフラグKONが1でなければ、今までキーオフで発音状態でなかったの  
で、ステップW6へ進み、キーオンフラグKONを1にして、ステップW7へ進む。

【0110】ステップW7では、前回のキーオフを示す休符の長さを示すデュレーションコードとデュレーションレジスタDの値を書き込む。書き込みを終了した後、演奏データメモリの書き込みポインタを進めて、ステップW8へ進む。

【0111】ステップW8では、演奏データメモリに今回のキーオンを示すキーオンコードとメモリMEM

(P)に格納されているキーコードを書き込む。書き込みを終了した後、演奏データメモリの書き込みポインタを進めて、ステップW9へ進む。

【0112】ステップW9では、演奏データメモリに書き込んだメモリMEM(P)のキーコードをレジスタKCに格納する。レジスタKCに格納されたキーコードは、次の配列番号P+1の処理において、同じキーコードが連続しているか否かの判断に用いられる。

【0113】ステップW10では、今回演奏データメモリにキーオンコードと共に書き込まれたキーコードの持続時間をカウントし直すために、デュレーションレジスタDを0にリセットする。その後、図10の処理へ戻り、次の配列番号P+1についての演奏データの生成処理を行う。

【0114】次に、前回演奏データメモリに書き込んだキーコードと同じキーコードがメモリMEM(P)に格納されている場合について説明する。つまり、メモリMEM(P)のキーコードとMEM(P-1)のキーコードが同じ場合である。

【0115】その場合、ステップW1において、メモリMEM(P)にオール1が格納されていないと判断され、さらにステップW2において、前回から引き続き今回もキーオン中であると判断され、ステップW3へ進む。

【0116】ステップW3では、メモリMEM(P)に格納されているキーコードとレジスタKCに格納されているキーコードが同じであるか否かを調べる。レジスタKCには、前回から引き続きキーオン中であるキーコードが格納されている。

【0117】メモリMEM(P)のキーコードとレジスタKCのキーコードが同じであれば、今回の配列番号Pについても前回と同じキーコードを引き続き発音することを示すので、演奏データメモリには何も書き込まずに、図10の処理へ戻り、デュレーションレジスタDの値のみをインクリメントして、キーコードの発音持続時

間をカウントする。

【0118】次に、前回演奏データメモリに書き込んだキーコードと異なるキーコードがメモリMEM(P)に格納されている場合について説明する。つまり、キーオン中において、発音すべきキーコードが変化する場合である。

【0119】その場合、ステップW1において、メモリMEM(P)にオール1が格納されていないと判断され、さらにステップW2において、前回から引き続き今回もキーオン中であると判断され、ステップW3へ進む。

【0120】ステップW3では、メモリMEM(P)に格納されているキーコードとレジスタKCに格納されているキーコードが同じであるか否かを調べる。メモリMEM(P)のキーコードとレジスタKCのキーコードが同じでなければ、今回の配列番号Pについての発音指示は前回のものとは異なるキーコードの発音を意味するので、ステップW4へ進む。

【0121】ステップW4では、演奏データメモリに前回までのキーコードの発音の持続時間を示すデュレーションレジスタDの値をデュレーションコードと共に書き込む。書き込みを終了した後、演奏データメモリの書き込みポインタを進めて、次の書き込みに備える。

【0122】ステップW5では、前回のキーコードについての発音終了を示すキーオフコードを演奏データメモリに書き込み、演奏データメモリの書き込みポインタを進める。

【0123】ステップW8では、今回のキーコードについての発音開始を示すキーオンコードとメモリMEM(P)に格納されているキーコードを演奏データメモリに書き込み、演奏データメモリの書き込みポインタを進める。

【0124】その後、ステップW9においてメモリMEM(P)に格納されているキーコードをレジスタKCに格納し、ステップW10においてデュレーションレジスタDを0にリセットし、次の発音持続時間をカウントし直す。そして、図10の処理に戻り、次の配列番号P+1についての処理を行う。

【0125】次に、メモリMEM(P)にキーコードでなく無音状態(休符)を示すオール1が格納されている場合の演奏データ生成手順を説明する。その場合、ステップW1において、メモリMEM(P)にオール1が格納されていると判断され、ステップW11に進む。

【0126】ステップW11では、キーオンフラグKONが1であるか否かを調べる。キーオンフラグKONが1であれば、キーオフを指示するためにステップW12へ進み、キーオンフラグKONを0にする。

【0127】ステップW13では、演奏データメモリに前回までのキーコードの発音の持続時間を示すデュレーションコードとデュレーションレジスタDの値を書き込

む。書き込みを終了した後、演奏データメモリの書き込みポインタを進めて、次の書き込みに備える。

【0128】ステップW14では、前回のキーコードについての発音終了を示すキーオフコードを演奏データメモリに書き込み、演奏データメモリの書き込みポインタを進める。その後、ステップW10へ進みデューションレジスタDを0にリセットして、キーオフ後の休符の持続時間をカウントし直す。そして、図10の処理へ戻り、次の配列番号P+1についての処理を行う。

【0129】メモリMEM(P)にオール1が格納されていると判断した後において、ステップW11でキーオンフラグKONが1でないと判断されたときには、キーオフの状態が前回から引き続いていることを示すので、演奏データメモリに何も書き込まずに図10の処理に戻り、キーオフ中である休符の持続時間を示すデューションレジスタDをインクリメントし、次の配列番号P+1についての処理を行う。

【0130】以上、演奏データの生成手順を述べたが、演奏データの形式は上述の形式に限られず、MIDI形式等の演奏データを生成するようにしてもよい。本実施例は、マイクから入力される音声信号に対して、量子化処理を行った後に平均ピッチを検出することにより、ピッチの変動に対して悪影響を受けることなく、正しいピッチを検出することができる。

【0131】また、検出されたピッチに対して、調検出により得られる調を用いて、誤認識により検出されたピッチの修正を行う。調によって決まる音楽規則を利用して、ピッチの修正処理を行うことにより、認識率の高いピッチ検出を実現することができる。検出修正されたピッチは、演奏データ変換処理を行うことにより、発音可能な演奏データに変換される。

【0132】なお、本実施例では、入力された音声から調を検出したが、スイッチ等を用いて演奏者が調を入力するようにしてもよい。以上実施例に沿って本発明を説明したが、本発明はこれらに制限されるものではない。例えば、種々の変更、改良、組合わせ等が可能なことは当業者に自明であろう。

【0133】

【発明の効果】入力された音声信号から検出した複数の時間区間毎のピッチについての平均ピッチを求めることにより、ピッチの細かな変動による悪影響を減らすこと

ができる。平均化されたピッチを基にして音高情報を生成することにより、正しい音高情報を得ることができる。

【0134】また、楽曲を構成する入力音声信号から検出された音高情報を基にして調の検出を行った後に、検出された調の音楽規則に応じて音高情報の修正を行う。音高情報の修正を行うことにより、正しい音高情報を得ることができる。

【図面の簡単な説明】

10 【図1】 本発明を実施するための音声信号変換装置のシステム構成例を示すブロック図である。

【図2】 入力変換処理のメインルーチンを示すフローチャートである。

【図3】 図2のステップP3における音声入力処理と平均ピッチ検出処理の詳細を示すフローチャートである。

【図4】 CPUが行う割り込み処理を示すフローチャートである。

20 【図5】 メモリMEM()の構成例を示す概念図である。

【図6】 図2のメインルーチン中のステップP6で行うキーコード変換の処理の詳細を示すフローチャートである。

【図7】 図2のメインルーチン中のステップP7の調内キーコードへの修正処理の詳細を示すフローチャートである。

【図8】 データメモリに格納されている調の構成音の音名データを示す概念図である。

30 【図9】 図7のステップT2の調内音丸め込み処理の詳細を示すフローチャートである。

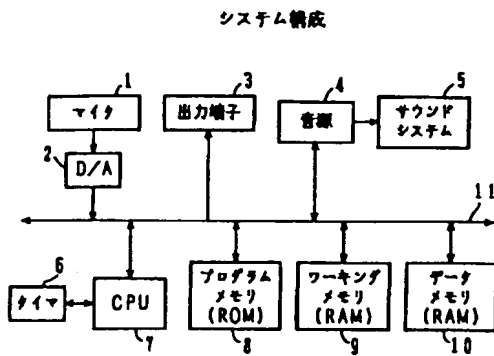
【図10】 図2のメインルーチン中のステップP8の演奏データ変換処理の詳細を示すフローチャートである。

【図11】 図10のステップV4における演奏データ変換処理の詳細を示すフローチャートである。

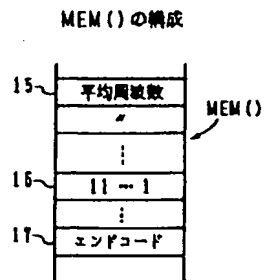
【符号の説明】

1 マイク、 2 D/A変換器、 3 出力端子、 4 音源、 5 サウンドシステム、 6 タイマ、 7 CPU、 8 プログラムメモリ、 9 ワーキングメモリ、 10 データメモリ、 11 バス

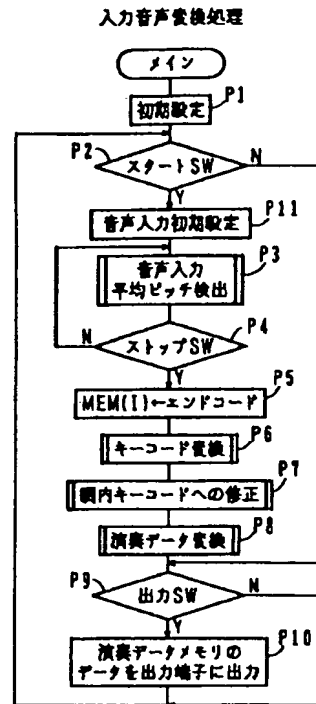
【図1】



【図5】

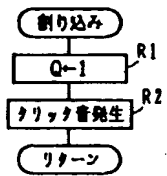


【図2】

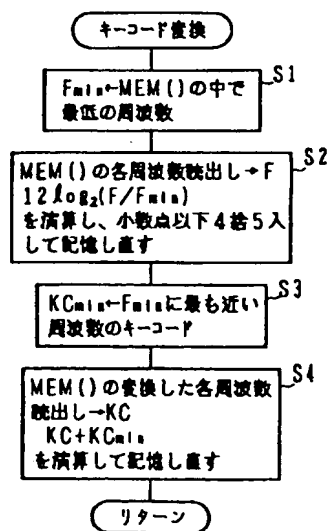


【図4】

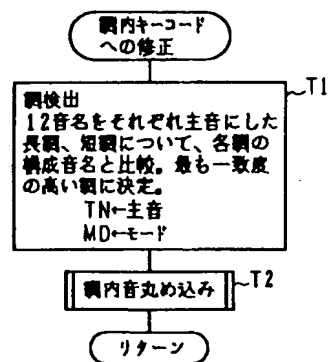
16分長毎の割り込み処理



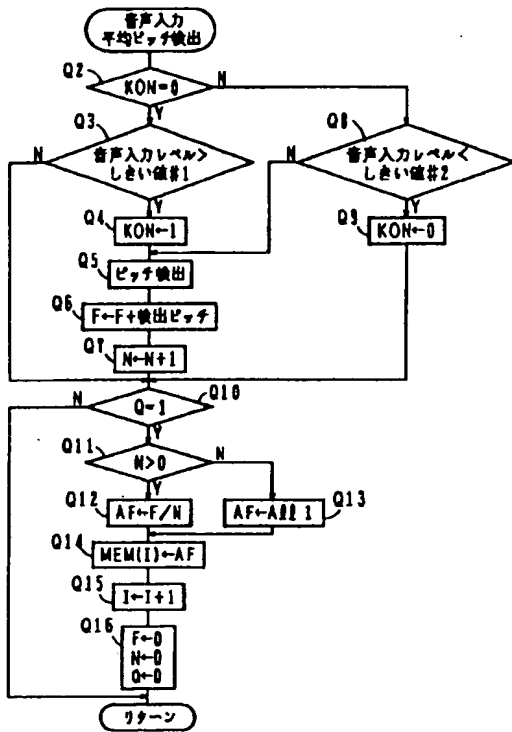
【図6】



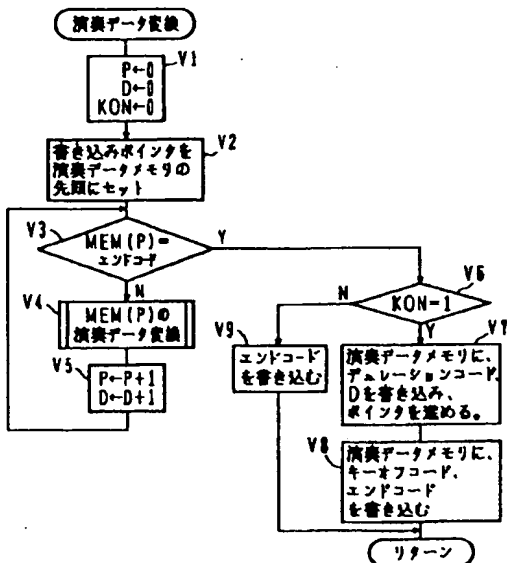
【図7】



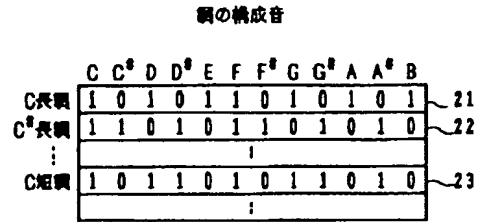
【図3】



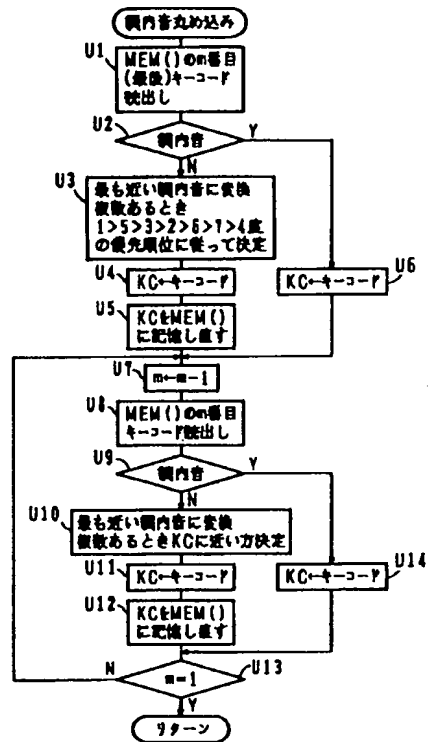
【図10】



【図8】



【図9】



【図11】

